Università degli Studi dell'Insubria
Facoltà di Scienza MM.FF.NN
Corso di laurea in Informatica

Realizzazione di uno strumento per la gestione ed organizzazione dei tempi di esecuzione di JTabwb

Tesi di Laurea di Eleonora Altana
Num. di matricola 609197
Relatore: Mauro Ferrari

Indice

Introduzione

Questo progetto nasce con lo scopo di gestire i confronti fra vari implementatori di prover su benchmark prestabililiti, provenienti dal framework JtabWb.

JTabWb è un framework Java per lo sviluppo di sperimentatori basati su sequenze di terminazioni successive o calcoli tableau.

Il framework è nato come strumento per la sperimentazione e il confronto di diversi calcoli e strategie proof-search per la logica proposizionale intuizionista.

Ora è un avanzato framework generale che può essere utilizzato per implementare diverse logiche e calcoli.

Può essere utilizzato per implementare provers come applicazioni Java stand-alone o come API da integrare in altre applicazioni Java. Diversamente dal altri framework come [1, 4, 5], in JTabWb l'utente può specificare tutti i componenti di una cella tra cui:

formule, nodi a prova di ricerca, regole e strategie.

Questo permette di implementare facilmente sperimentatori per diverse logiche e calcoli diversi (sequent stile o tableau in stile calculi).

Il suo limite principale è che tutti i componenti sono forniti come classi Java, quindi l'utente deve avere esperienza con questo linguaggio.

La natura object oriented del linguaggio e la composizione del framework, permette il riutilizzo dei componenti di una cella.

Questo permette di sviluppare facilmente diverse varianti di un prover, in modo da confrontare diverse implementazioni di strutture dati (formule, sequenti,...) e diverse strategie.

Il framework fornisce anche il supporto per la generazione di proof-traces (storie di proof-search), il rendering di prove LATEX e generazione counter-model.

JTabWb e alcuni sperimentatori per logica intuizionista proposizionale implementata, sono disponibili all'indirizzo http://www.dista.uninsubria. it/~ferram

Le informazioni provenienti dal sistem JTabWb contengono dati riguardanti l'esecuzione di problemi all'interno del sistema, l'indicazione che questi problemi sono risolvibili, il tempo di esecuzione, la famiglia di appartenenza, il tipo di prover di esecuzione.

Questo progetto realizza uno strumento capace di immagazzinare i dati provenienti da

questo framework, creare un'anagrafica delle macchine di esecuzione dei calcli e permette di analizzare e confrontare, attraverso strumenti visivi come i grafici, quali famiglie di prover si comportino meglio su determinate macchine, siano peggiori in alcuni ambienti, abbiamo tempi di esecuzione molto alti, etc.

Questi confronti sono effettuati su testset di due tipologie: SYJ e RANDXX.

I due testset hanno comportamenti differenti, quindi l'analisi delle famiglie di prover appartenenti a questi testset è differente:

nel primo caso abbiamo un confronto dei tempi di esecuzione, mentre nel secondo valitiamo l'andamento medio.

Vedremo nei capitoli seguenti in dettaglio come sono effettuati questi confronti.

Progetto

Il progetto e stato realizzato utilizzando il linguaggio di programmazione Java8 e le sue librerie integrate JavaFX per implementare la parte di layout dell'applicazione.
La scelta di utilizzare questo linguaggio è stato fatta per avere un modo di conoscere meglio questo linguaggio, da me poco utilizzato, uno dei piu' diffusi al mondo.
Parleremo meglio delle tecnologie utilizzate nella sezione Tecnologie utilizzate.

Funzionamento

All'avvio, l'applicazione presenta direttamente la pagina di confronto dei teoremi, la quale contiene le funzionalita' di principale interesse ai fini del progetto.

Parleremo in dettaglio della pagina di comparazione nei paragrafi seguenti.

Alla prima esecuzione, l'applicazione non contiene alcun tipo di dato, presentando la pagina di comparazione completamente vuota.

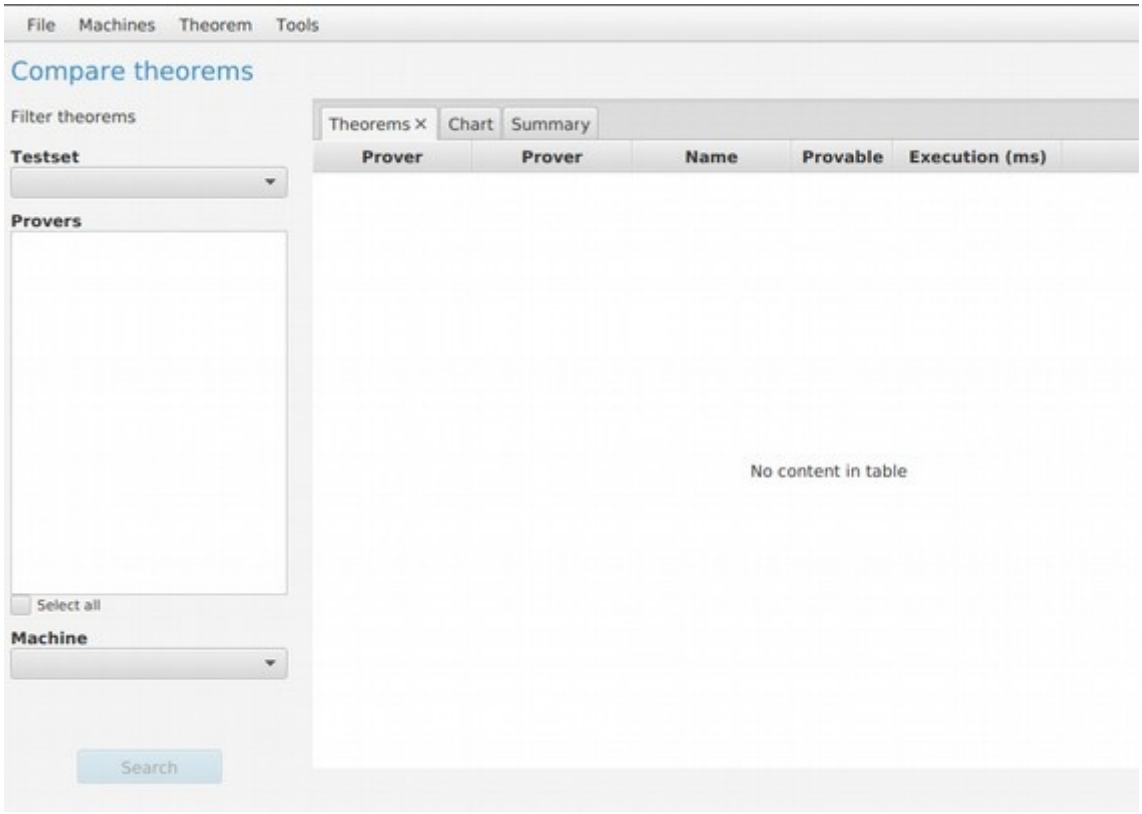


Illustrazione 1: Pagina di comparazione

Per avere dei dati da poter comparare, è necessario censire un'anagrafica delle macchine e caricare almeno uno o piu' file contenenti tempi di esecuzione.

Anagrafica macchine

Per gestire l'anagrafica delle macchine, basta selezionare la voce “List” dal menu “Machine”

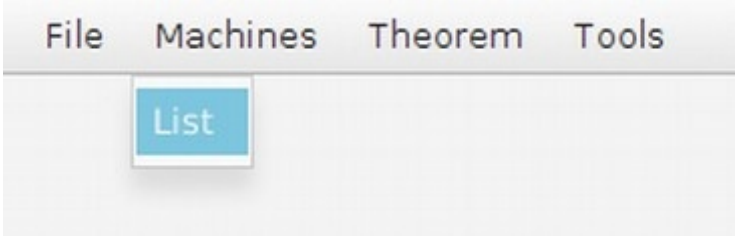


Illustrazione 2: Menu macchine

e si accede alla pagina dedicata, dove viene visualizzata la seguente schermata:

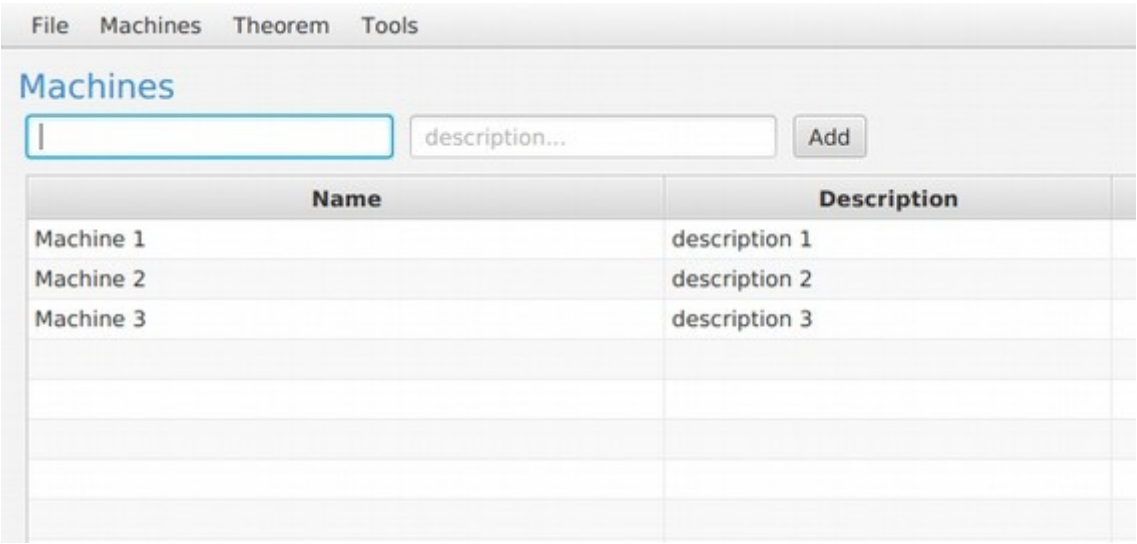


Illustrazione 3: Pagina di gestione macchine di esecuzione

Nuova macchina

Inserendo un nome da assegnare alla macchina (cella selezionata nella figura) ed una descrizione, basta cliccare sul pulsante “Add” ed il sistema inserisce nel database le informazioni.

La tabella delle macchine, presente nel database, è stata creata con un indice di univocita' della colonna nome, per impedire che venga inserita piu' volte uno stesso nome macchina. L'applicazione sfrutta questa caratteristica del database, gestendo l'eventuale eccezione che la libreria `sqlite-jdbc` (che si occupa della comunicazione con il database) genera: quando viene intercettata l'eccezione `java.sql.SQLException`, l'applicazione visualizza all'utente la seguente finestra di dialogo

Illustrazione 4: Errore inserimento macchina

per indicare che l'operazione di salvataggio non è andata a buon fine.

Modifica macchina

Se si desidera cambiare il nome o la descrizione di una macchina presente nel sistema, basta selezionare la riga dalla tabelle che le elenca ed effettuare doppio click sulla cella che si desidera modificare. In questo modo la cella diventa editabile ed è possibile modificarne il contenuto, come visibile nella figura qui sotto

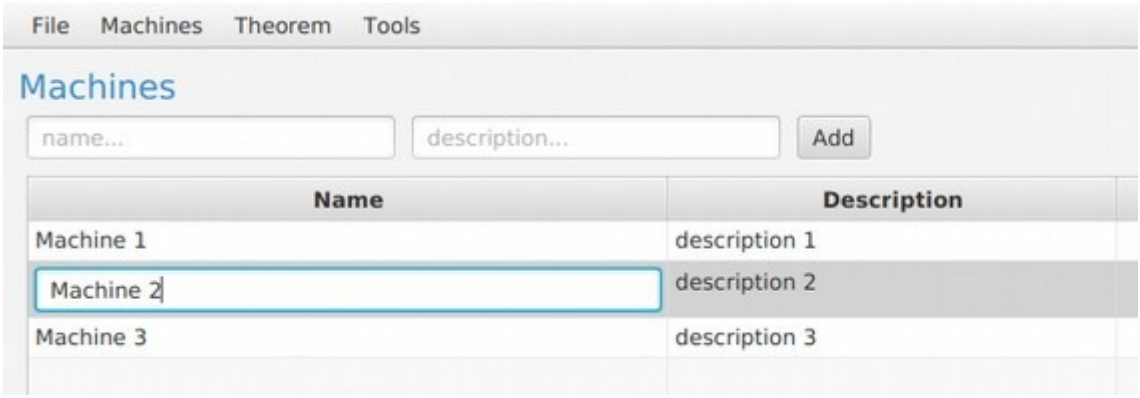


Illustrazione 5: Modifica informazioni macchina

Dati di esecuzione

Caricamento

è possibile visualizzare la pagina per il caricamento dei file, selezionando la voce “Upload” del menu “Theorems”

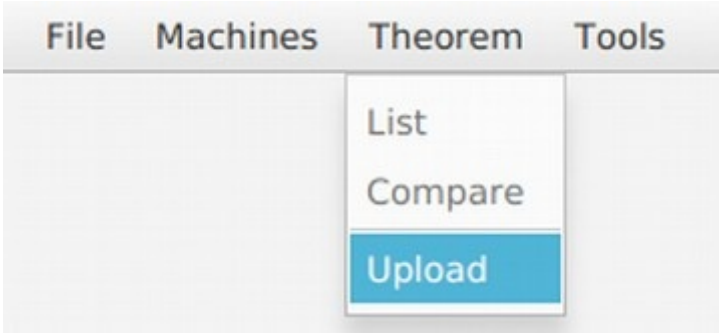


Illustrazione 6: Menu caricamento file

La pagina di caricamento presenta una ComboBox popolata con le anagrafiche macchine, fra cui l'utente deve scegliere

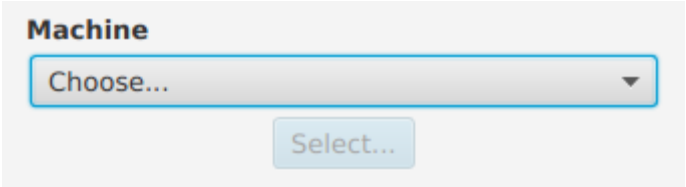


Illustrazione 7: Scelta della macchina

come si puo' vedere nell'illustazione 7, il pulsante denominato con “Select...” non è abilitato e non puo' essere cliccato: questa scelta di visualizzazione impedisce all'utilizzatore di poter caricare uno o piu file senza che il sistema possa salvare nel database teoremi senza l'indicazione della macchina in cui sono stati eseguiti.

Selezionando una macchina dall'elenco, il pulsante si abilita

Illustrazione 8: Macchina selezionata, caricamento abilitato

e permette la selezione dei files, visualizzando un FileChooser

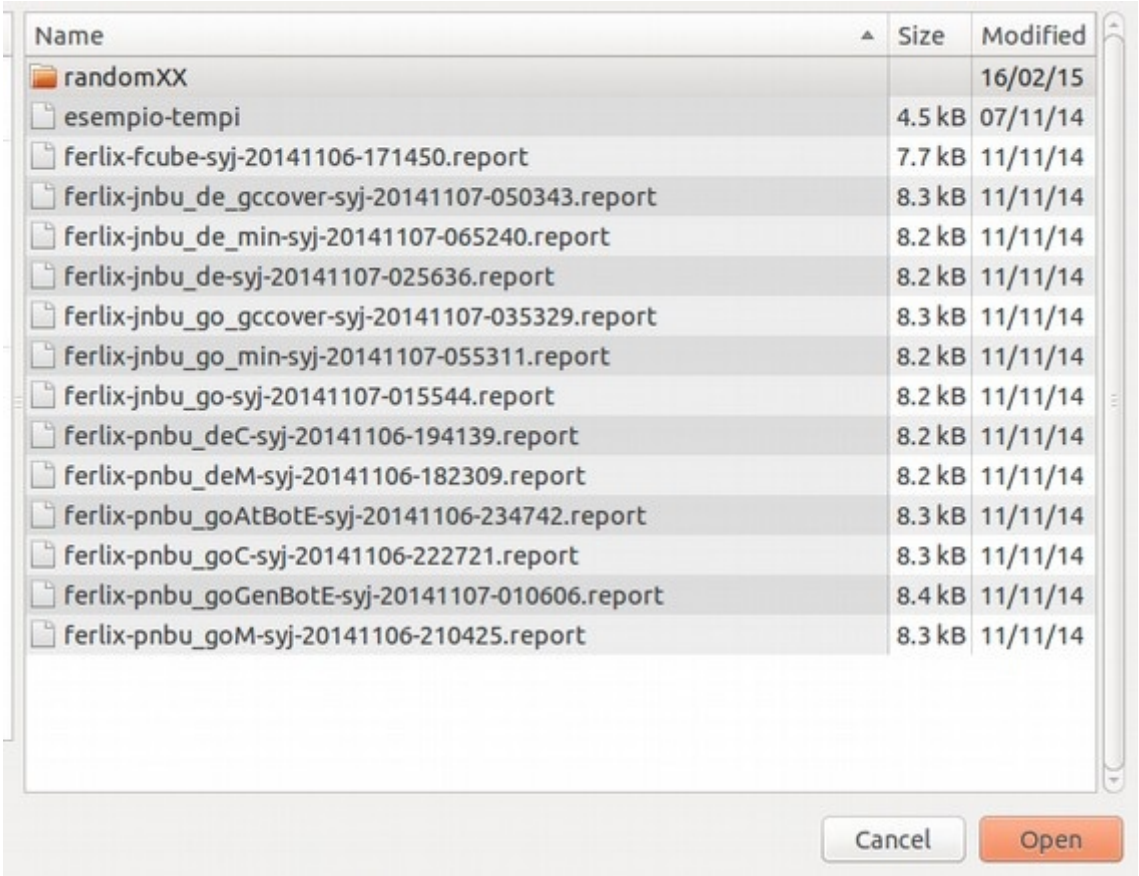


Illustrazione 9: FileChooser per la selezione dei file da caricare

è possibile selezionare uno o piu' file alla volta e, una volta fatto, il sistema procede a caricarli uno alla volta nel sistema.

Questo caricamento potrebbe richiede diverso tempo di esecuzione, quindi la procedura viene eseguita in un thread parallelo all'applicazione, in modo da impedire che l'applicazione si possa bloccare.

```
Task<Void> task = new Task<Void>() {
    @Override
    public Void call() {
        Integer machine_id = machineCombo.getValue().id;
        long filesCount = files.size();
        for (File file : files) {
            try {
                TheoremParser.processFile(file, machine_id);
```

```
            } catch (IOException e) {
                this.cancel(true);
                loadingIndicator.setVisible(false);
                showErrorMessage("Error during file reading:"
                                 + e.toString());
            } catch (SQLException e) {
                this.cancel(true);
                loadingIndicator.setVisible(false);
                showErrorMessage("Error during database operation:"
                                 + e.toString());
            } catch (Exception e) {
                this.cancel(true);
                loadingIndicator.setVisible(false);
                showErrorMessage("Error:"+ e.toString());
            }
            updateProgress(
                 (long) files.indexOf(file) + 1, filesCount);
        }
        return null;
    }
```

(Il concetto di thread verra' approfondito nella sezione Threads)

L'applicazione “segue” l'avanzamento del thread, attraverso una ProgressBar dal nome loadingIndicator, collegando la property progress del task a quella della progress bar

`loadingIndicator.progressProperty().bind(task.progressProperty());`
visualizzando cosi lo stato dell'esecuzione.

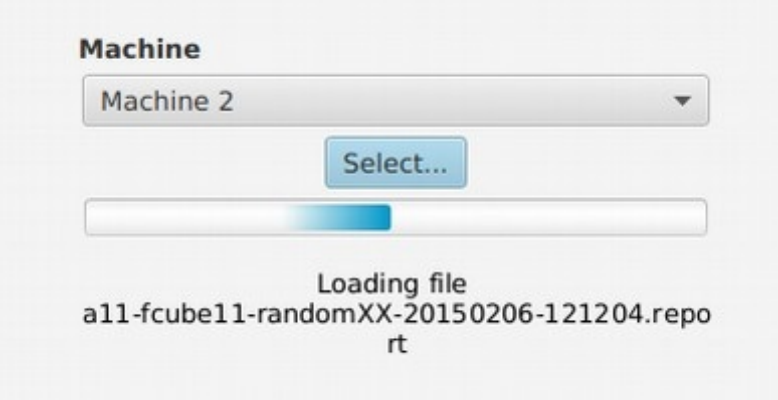


Illustrazione 10: Feedback dell'applicazione durante il caricamento file

Una Label che informa l'utente di quale file il sistema stia processando, come si vede nell'illustrazione 10.

Una volta terminata la procedura di caricamento, l'applicazione visualizza un messaggio di conferma

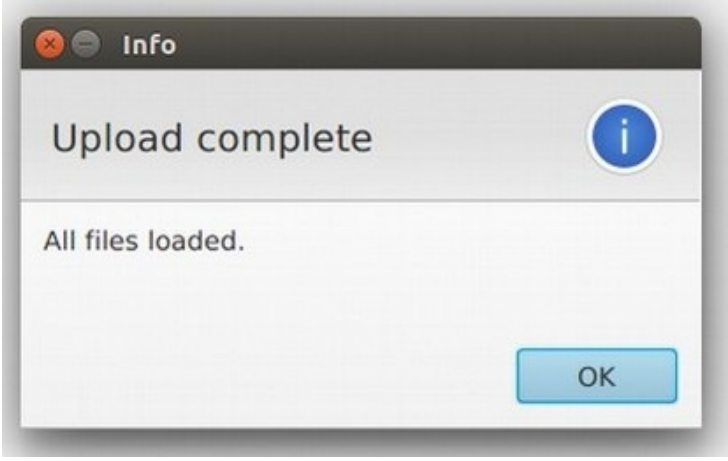


Illustrazione 11: Conferma procedura di caricamento completata

Elenco

è possibile visualizzare l'elenco dei teoremi presenti nel database del sistema.

Per farlo bisogna selezionare la voce “List” dal menu “Theorems”.

Possiamo utilizzare questa pagina anche per effettuare una verifica dei dati che sono stati caricati, in modo da controllare di avere le informazioni che ci interessano.

La pagina presenta in una TableView, tutti i teoremi presenti nel sistema

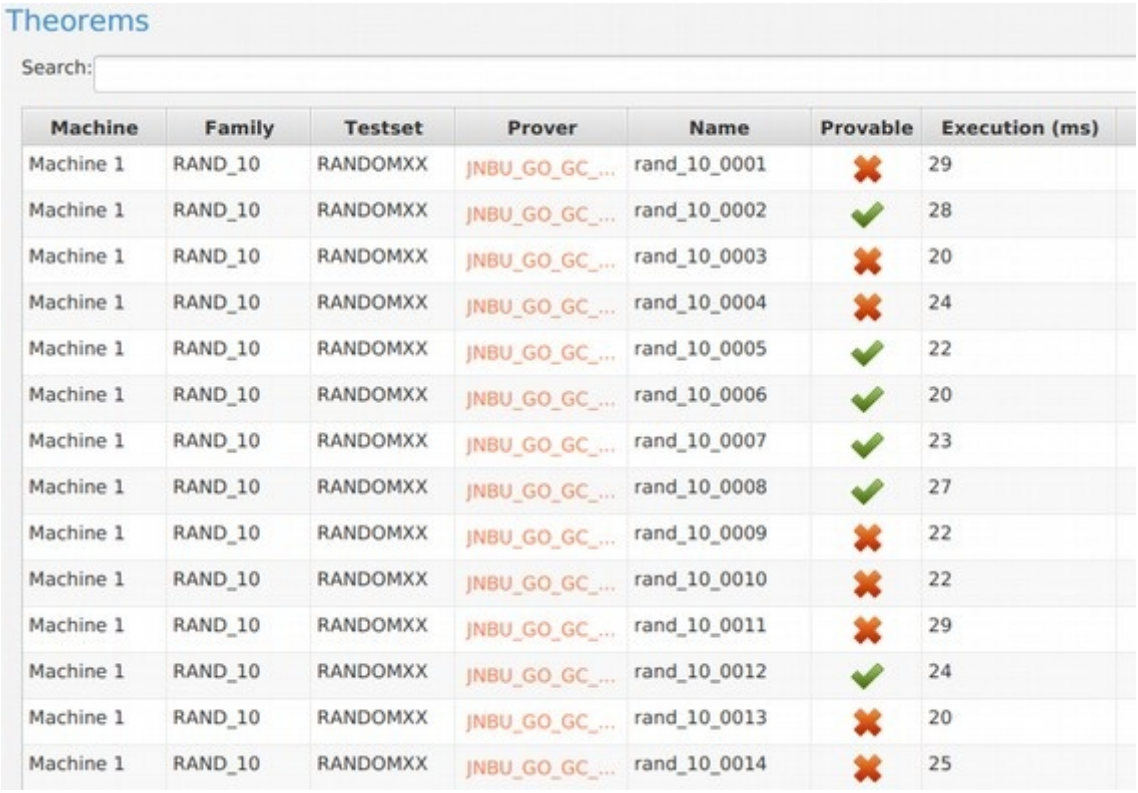


Illustrazione 12: Elenco completi teoremi del sistema

l'elenco è ordinabile cliccando l'intestazione delle colonne, ed è possibile ricerca uno specifico teorema attraverso la `TextBox` “Search”, in modo da filtrare l'elenco dei teoremi.

La ricerca è solamente testuale e ricerca all'interno di tutti i valori del teorema.

Comparazione

Dopo aver inserito le anagrafiche macchine e popolato il database con i dati di esecuzione, possiamo procedere alla sezione di comparazione.

Si inizial col selezionare i parametri di filtro per ricercare i problemi eseguiti di cui si vuole fare un confronto, i parametri di ricerca sono:

- Macchina di esecuzione (Machine)
- Testset (SYJ o RANDOMXX)

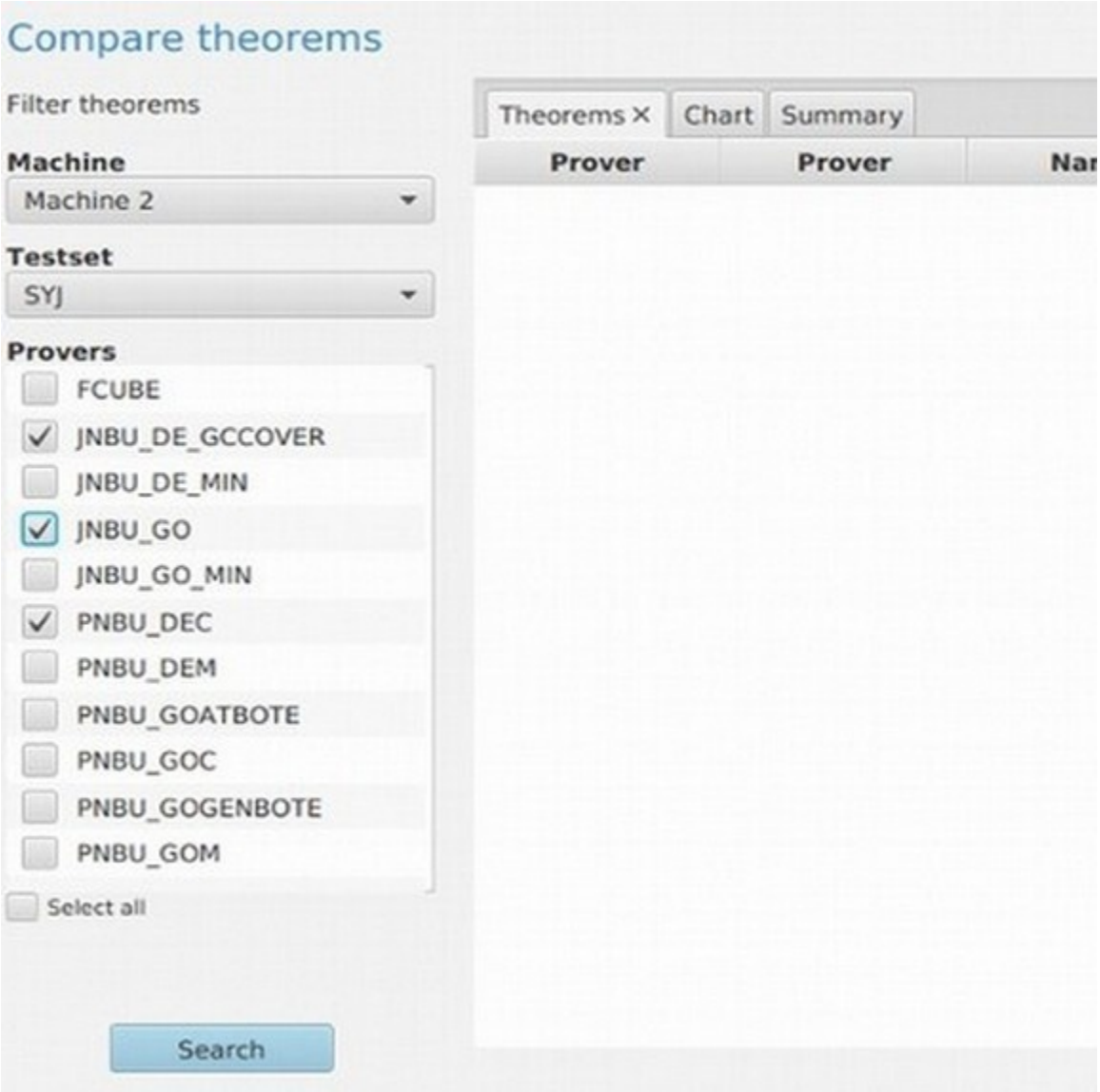


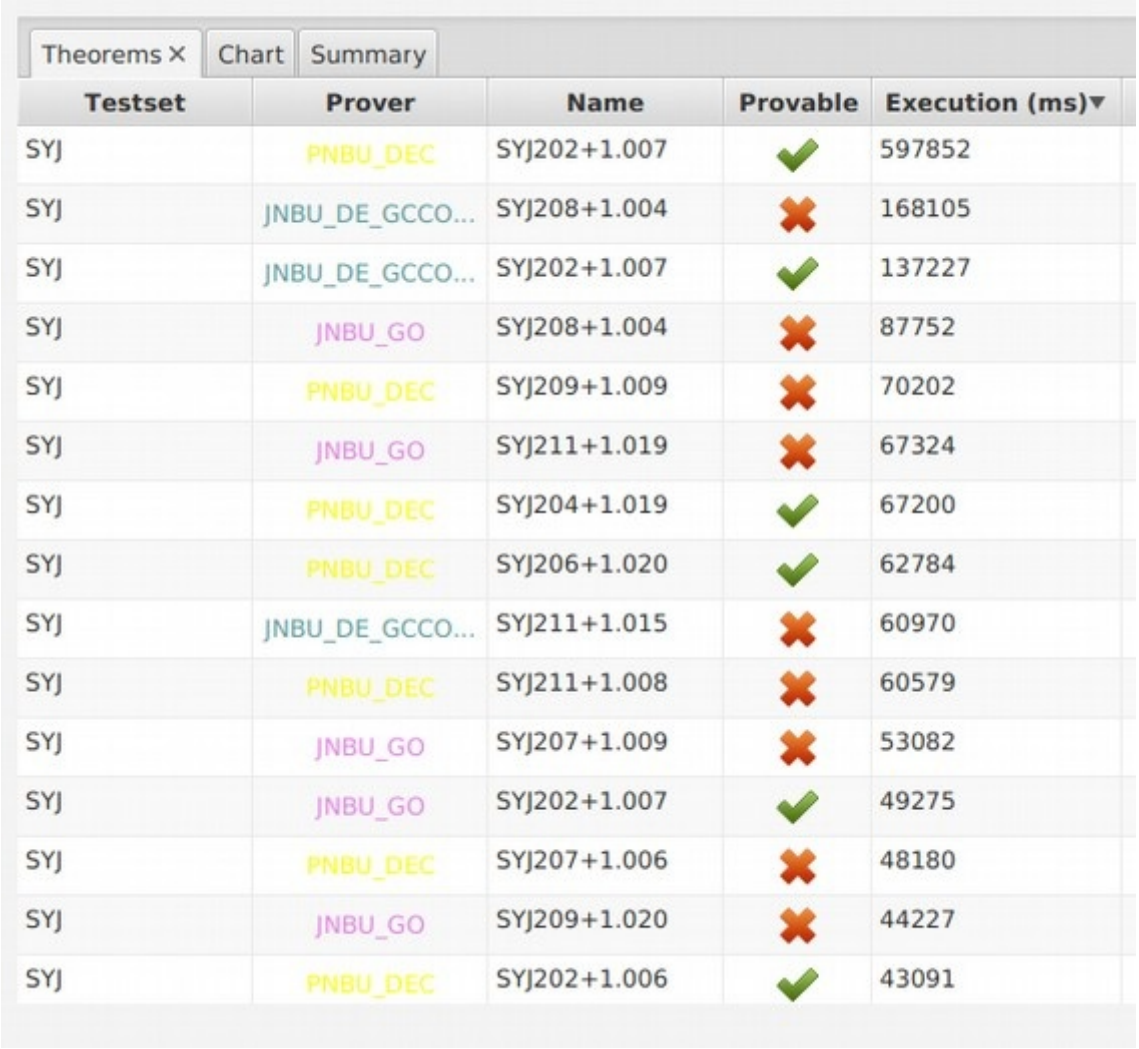
Illustrazione 13: Selezione dei parametri di filtro per la comparazione

questi due criteri servono per ricercarea all'interno del dabatase e popolare l'elenco della ListView “Provers” con l'elenco dei prover eseguiti sulla macchina scelta e che

appartengono al testset selezionato.

A questo punto, selezionando dall'elenco provers quelli desiderati, possiamo procedere alla ricerca dei dati, cliccando il pulsante “Search”. Il sistema ricerca i dati che corrispondono ai criteri selezionati, visualizzando nel tab “Theorems” l'elenco dei risultati della ricerca.

La TableView visualizza l'elenco dei teoremi presenti nel database, indicando il testset, il nome del prover, evidenziando i diversi nomi con colori differenti, il nome del teorema, l'indicazione che indica se il teorema è provable e quanti millisecondo ha impiegato ad essere eseguito. La TableView possiede tutte le colonne ordinabili, quindi è possibile scegliere l'ordinamento che si desidera.


Theorems × | Chart | Summary

Testset	Prover	Name	Provable	Execution (ms)▾
SYJ	PNBU_DEC	SYJ202+1.007	✓	597852
SYJ	JNBU_DE_GCCO...	SYJ208+1.004	✗	168105
SYJ	JNBU_DE_GCCO...	SYJ202+1.007	✓	137227
SYJ	JNBU_GO	SYJ208+1.004	✗	87752
SYJ	PNBU_DEC	SYJ209+1.009	✗	70202
SYJ	JNBU_GO	SYJ211+1.019	✗	67324
SYJ	PNBU_DEC	SYJ204+1.019	✓	67200
SYJ	PNBU_DEC	SYJ206+1.020	✓	62784
SYJ	JNBU_DE_GCCO...	SYJ211+1.015	✗	60970
SYJ	PNBU_DEC	SYJ211+1.008	✗	60579
SYJ	JNBU_GO	SYJ207+1.009	✗	53082
SYJ	JNBU_GO	SYJ202+1.007	✓	49275
SYJ	PNBU_DEC	SYJ207+1.006	✗	48180
SYJ	JNBU_GO	SYJ209+1.020	✗	44227
SYJ	PNBU_DEC	SYJ202+1.006	✓	43091

Illustrazione 14: Elenco risultati ricerca

Chart SYJ

Passando alla sezione “Chart”, abbiamo una rappresentazione visiva del confronto del testset SYJ. Per questo testset, il confronto che vogliamo eseguire è sapere sia stato il prover dove i problemi sono stati risolti piu velocemente.

Questo confronto lo facciamo utilizzando un grafico a barre, come mostrato nell'illustrazione 15

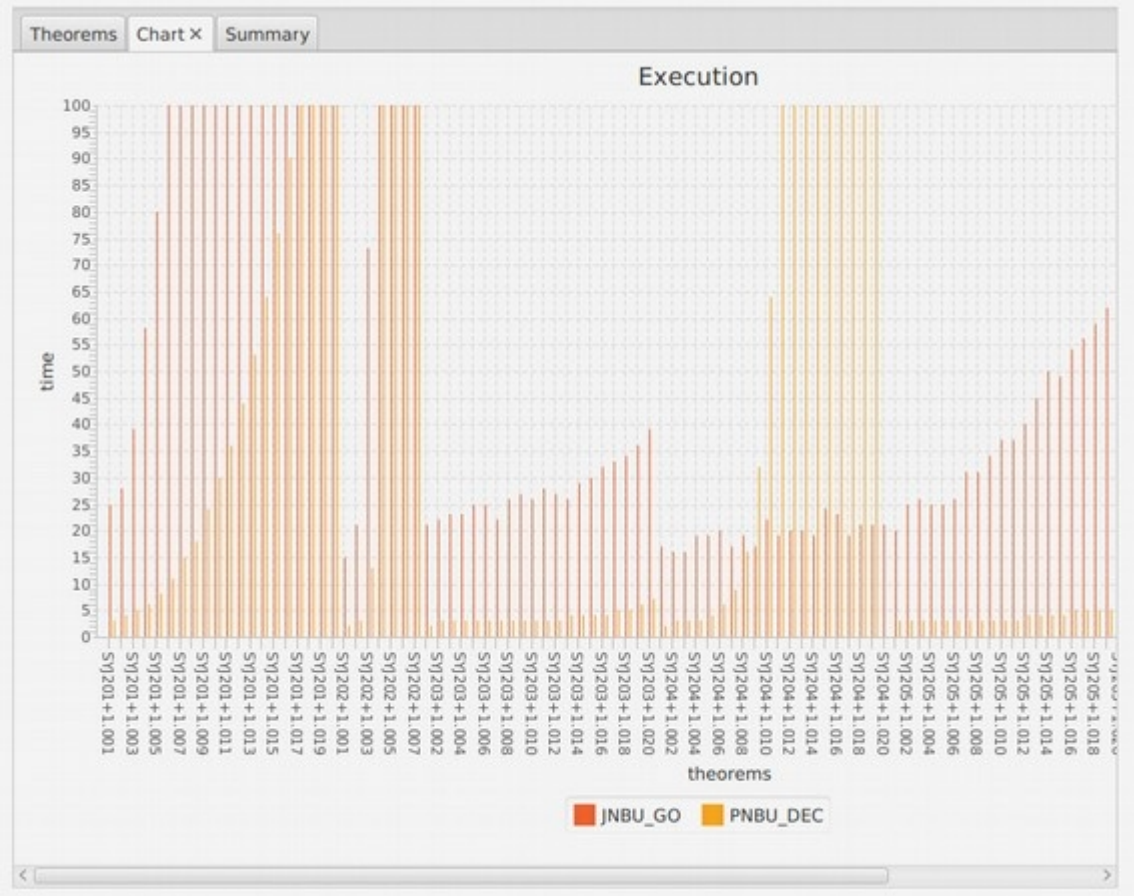


Illustrazione 15: Grafici di comparazione testset SYJ

Vista la numerosita' di questo grafico e le possibili ampie differenze dei tempi di esecuzione, è stata implementata una classe `ZoomHandler` per permette di effettuare lo zoom del grafico: basta tenere premuto il pulsante `Ctrl` mentre si fa scroll con il mouse, fino a che non si arriva al livello di dettaglio di confronto che si desidera.

Dalla visualizzazione in dettaglio del grafico, possiamo vedere in modo immediato come uno stesso problema si comporti meglio se eseguito con un prover piuttosto che in un altro: nel caso che vediamo nell'illustrazione 16, il problema `SYJ201_1.001` viene eseguito piu velocemente se eseguito su prover `PNBU_DEC` (in giallo) che se eseguito con prover `JNBU_GO` (in arancione).

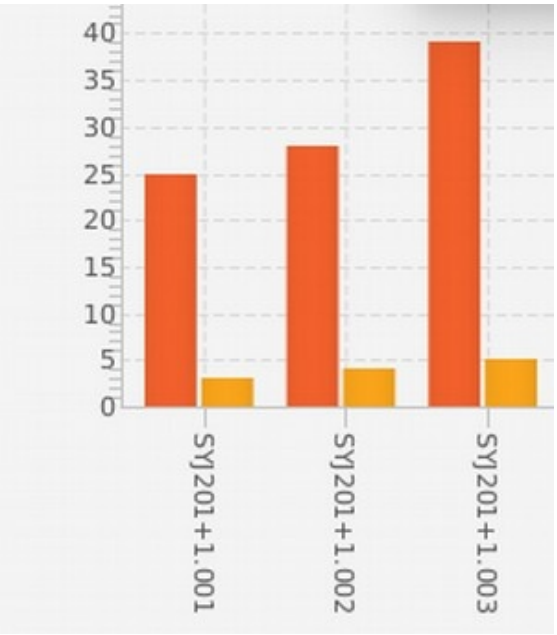


Illustrazione 16: Zoom dettaglio grafico di comparazione SYJ

Chart RANDXX

Per questo tipo di comparazione, utilizziamo un grafico a linea, come visualizzato nell'illustrazione 17.

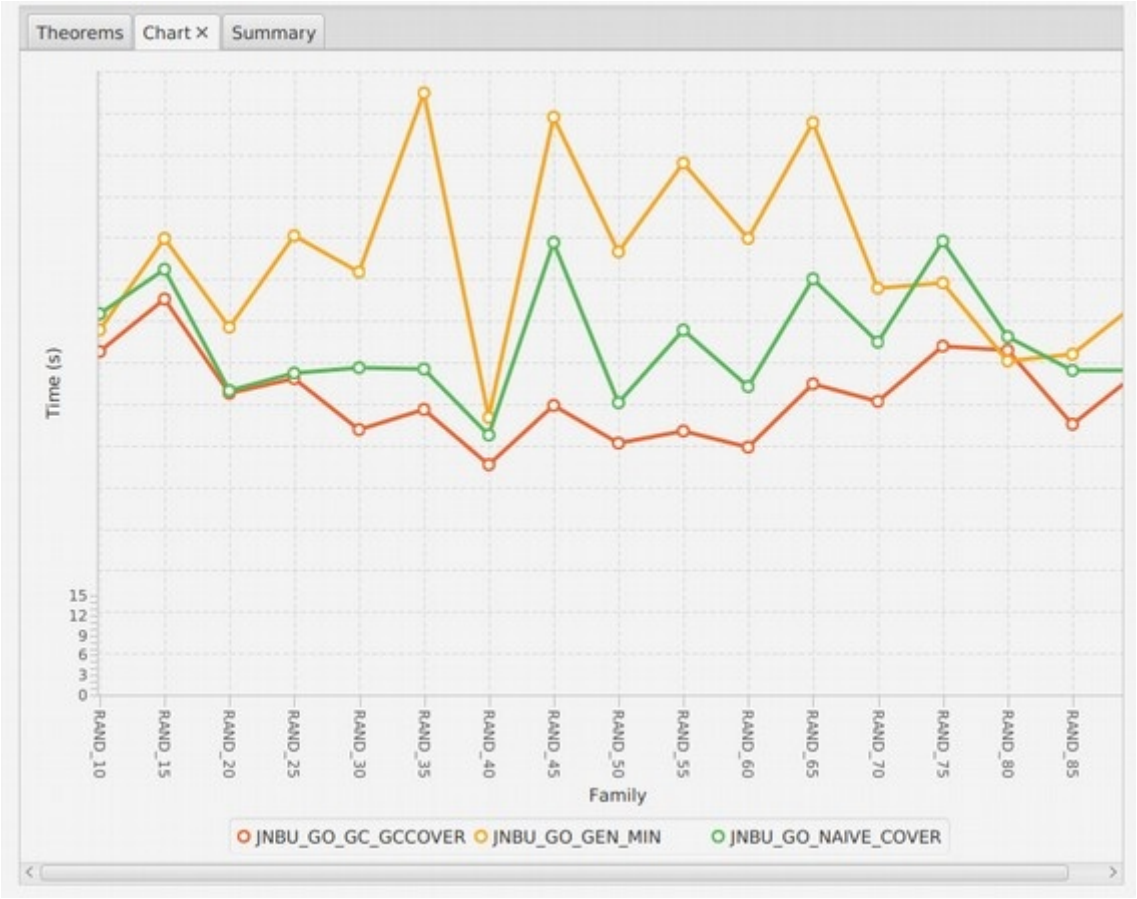


Illustrazione 17: Grafico di comparazione testset Randxx

In questo caso il tipo di confronto riguarda il tempo medio di esecuzione di problemi per ogni famiglia di problemi appartenente ai prover selezionati.

Il valore medio è calcolato secondo la formula

```
(t1 + t2 ) / totale_istanze
```

dove

T1 è il tempo impiegato per risolvere tutte le istanze risolte

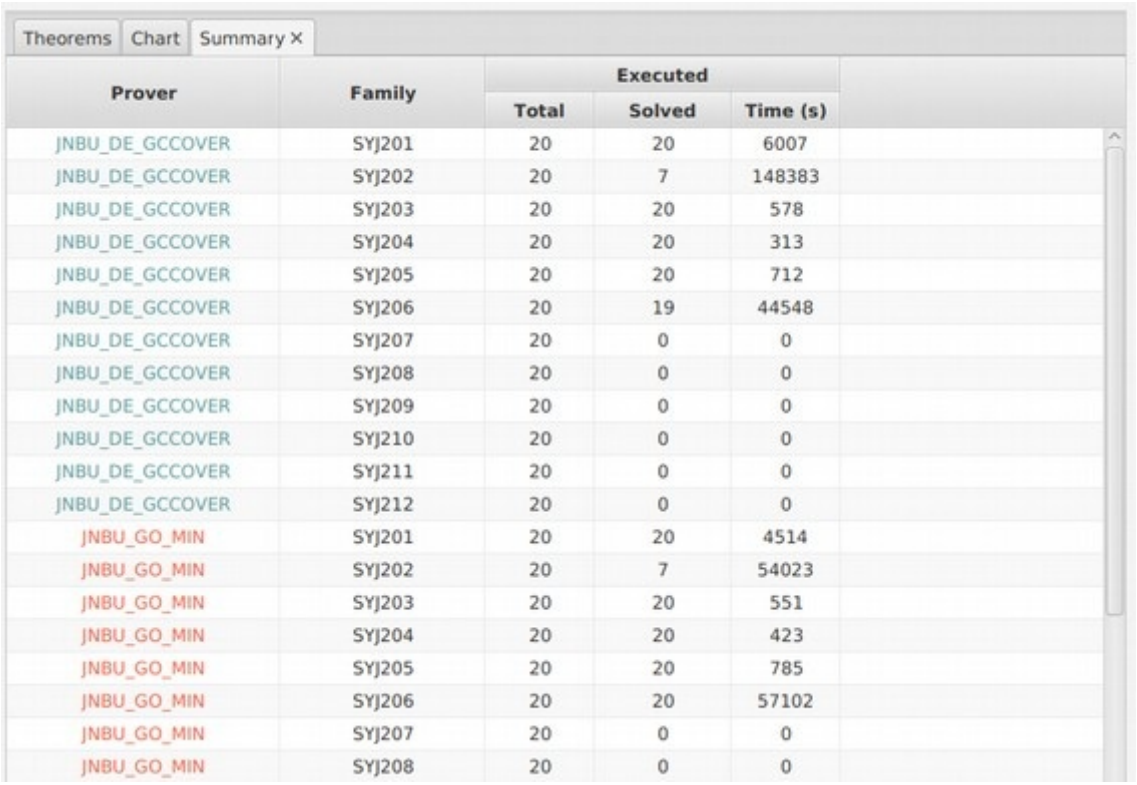
T2 rappresenta il calcolo (numero di instanze andate in timeout / timeout)

Il valore timeout viene indicato nell'intestazione del file: questa informazione viene salvata nel database come valore di timeout del problema.

Summary

Nella sezione sommario, troviamo i dati riepilogativi del confronto.

Questa sezione visualizza i dati utilizzati per i grafici di confonto, mantenendo quindi una stessa visualizzazione, ma applicando la logica utilizzata nei grafici: tutti i tempi di esecuzione nel caso del testset SYJ, i valori medi nel caso del testset RANDXX.


Theorems | Chart | Summary ×

Prover	Family	Executed		
		Total	Solved	Time (s)
JNBU_DE_GCCOVER	SYJ201	20	20	6007
JNBU_DE_GCCOVER	SYJ202	20	7	148383
JNBU_DE_GCCOVER	SYJ203	20	20	578
JNBU_DE_GCCOVER	SYJ204	20	20	313
JNBU_DE_GCCOVER	SYJ205	20	20	712
JNBU_DE_GCCOVER	SYJ206	20	19	44548
JNBU_DE_GCCOVER	SYJ207	20	0	0
JNBU_DE_GCCOVER	SYJ208	20	0	0
JNBU_DE_GCCOVER	SYJ209	20	0	0
JNBU_DE_GCCOVER	SYJ210	20	0	0
JNBU_DE_GCCOVER	SYJ211	20	0	0
JNBU_DE_GCCOVER	SYJ212	20	0	0
JNBU_GO_MIN	SYJ201	20	20	4514
JNBU_GO_MIN	SYJ202	20	7	54023
JNBU_GO_MIN	SYJ203	20	20	551
JNBU_GO_MIN	SYJ204	20	20	423
JNBU_GO_MIN	SYJ205	20	20	785
JNBU_GO_MIN	SYJ206	20	20	57102
JNBU_GO_MIN	SYJ207	20	0	0
JNBU_GO_MIN	SYJ208	20	0	0

Illustrazione 18: Tabella sommario testset SYJ

Questa TableView rappresenta il nome del prover, la famiglia di appartenenza e i valori di esecuzione dei problemi appartenenti a quella famiglia : totale dei problemi, numero di quelli risolvibile il tempo totale di esecuzione dei problemi risolvibili.

Tracciato file delle esecuzioni

Il sistema JTabWb fornisce una serie di file che rappresentano le esecuzioni dei problemi su diversi prover.

I file sono composti da una sezione di intestazione, che fornisce informazioni di carattere generale sui problemi contenuti in esso, come ad esempio il nome del prover, il testset di appartenenza e informazioni generali sull'esecuzione, come il totale delle istanze eseguite, il timeout massimo, il tempo totale di esecuzione, etc.

Ecco un esempio di intestazione di un file SYJ

```
%----------------------------------------
% Prover               : fcube
%                        FCube4
% Testset              : SYJ
% Timeout (sec)        : 600
% Total instances      : 240
% Solved instances     : 228
% Provable instances   : 108
% Unprovable instances: 120
% Errors               : 0
% Timeouts             : 12
% Total time (sec)     : 386.563
%
% Families: {
%    SYJ201, solved 20/20, time 2.299
%    SYJ202, solved 8/20, time 158.639
%    SYJ203, solved 20/20, time 0.89
%    SYJ204, solved 20/20, time 0.26
%    SYJ205, solved 20/20, time 0.64
%    SYJ206, solved 20/20, time 0.0
%    SYJ207, solved 20/20, time 1.148
%    SYJ208, solved 20/20, time 223.995
%    SYJ209, solved 20/20, time 0.96
%    SYJ210, solved 20/20, time 0.36
%    SYJ211, solved 20/20, time 0.42
%    SYJ212, solved 20/20, time 0.129
% }
%
```

Illustrazione 19: Intestazione file SYJ

Le informazioni che il sitema interpreta e salva sono:

- Prover – nome del prover (es fcube)
- Testset – nome del testset (SYJ)
- Timeout – valore di timeout massimo

Le altre informazioni riguardano il riepilogo delle informazioni riguardanti le righe di dettaglio. Questi dati sono stati usati solo per controllare le informazioni dopo aver effettuato l'upload dei file, in modo aver conferma che nel database sono stati salvati correttamente i dati.

Nelle righe di dettaglio, sono contenute le informazioni specifiche del problema eguito, come il nome e i dettagli di esecuzione (eseguibile, tempo di esecuzione, timeout)

Esempi di righe di dettaglio

1. `SYJ202+1.007; provable; 48783` - problema eseguibile

   in questo caso il problema viene salvato nel database come problema risolvibile in 48783 millisecondi

2. `SYJ202+1.008; unknown; timeout (600)` – problema in timeout

   il problema viene salvato nel database come non risolvibile, andato in timeout

3. `SYJ209+1.006; unprovable; 60` – problema non eseguibile

   in questo caso salviamo che il problema non è eseguibile e il tempo di esecuzione di 60 millisecondi

Struttura

Per la realizzazione di questo progetto, si è cercato di utilizzare il pattern MVC (Model-View-Controller)

Questo pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il **model** fornisce i metodi per accedere ai dati utili all'applicazione;
- il **view** visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
- il **controller** riceve i comandi dell'utente (attraverso il view) e li attua modificando lo stato degli altri due componenti.

In questo contesto l'mvc definisce anche la struttura dei package Java.

Package Model

In questo package sono racchiuse la classi model che vengono utilizzate dall'applicazione per rappresentare le informazioni contenute nel database.

Queste classi vengono utilizzate per rappresentare i dati contenuti nel database in un formato che i controlli UI, come TableView o ComboBox, possono interpretare per visualizzare le informazioni.

Le classi contenute nel package sono:

- `Machine` – racchiude le informazioni della macchina dove sono stati eseguiti i problemi
- `MachineTable` – rappresenta la classe utilizzata per visualizzare le informazioni macchina in una TableView. A differenza della classe Machine, contiene i metodi getter e setter che servono all'oggetto TableView per funzionare
- `Prover` – rappresenta le informazioni dell'oggetto prover (nome)
- `Testset` – rappresenta le informazioni del testste (nome)
- `Theorem` – rappresenta tutte le informazioni del problema (teorema eseguito) come tempo di esecuzione, nome, famiglia, testset, timeout
- `SummaryTable` – viene utilizzata per la rappresentazione delle informazioni della sezione di confronto dell'applicazione e contiene informazioni riassuntive sulle esecuzioni dei problemi (numero di teoremi totali, teoremi eseguiti, tempi

totali di esecuzione)

Package Controller

Questo package contiene tutti i controller utilizzati per gestire le varie pagine dell'applicazione.

Le classi controller si occupano di gestire le interazioni dell'utente con l'applicazione e di gestire le interazione fra tutte le entita' necessarie per leggere o scrivere informazioni sul database

La classe principale è `BaseController`, che definisce i metodi comuni a tutti i controller, come la visualizzazione di messaggi di feedback verso l'utente a seguito di un'azione, come la pressione di un pulsante.

Tutte le altre classi controller ereditano da BaseController, gestendo in maniera specifica una certa pagina dell'applicazione.

Le classi controller presenti sono:

- `AllTheoremController` si occupa di gestire l'interazione con la pagina di elenco di tutti i teoremi presenti. Gestisce l'evento ricerca e la visualizzazione dell'elenco teoremi attraverso l'oggetto TableView
- `CompareController` è il controller piu' importante dell'intera applicazione, in quando gestisce la pagina di comparazione dei teoremi. Questo controller si occupa di visualizzare i testset presenti nel database,i prover e l'elenco delle macchine di esecuzione, punto di partenza per effettuare la ricerca dei teoremi nel database per poi visualizzarli in elenco. Questo controller gestise inoltre la creazione e visualizzazione dei grafici di comparazione fra i prover, in modo da permettere un immediato confronto fra le parti.
- `MachineController` gestisce tutte le interazioni con la tabella delle macchine di esecuzione. Legge l'elenco delle macchine e permette di inserire e modificare le informazioni
- `MainController` si occupa di gestire la navigazione all'interno dell'applicazione attraverso le azioni sul menu. Il suo compito è quello di visualizzare la pagina a seconda della richiesta dell'utente, per poi delegarne il controllo al controller assegnato a tale pagina
- `TheoremController` gestisce il caricamento dei file dei teoremi all'interno del sistema. è possibile caricare anche piu' file alla volta: il caricamento viene effettuato atraverso l'esecuzione di un thread parallelo a quello dell'applicazione in modo da impedire che l'applicazione possa bloccarsi in attesa che il compito

venga svolto, soprattutto nel caso di caricamento di file che contengono molte righe di dettaglio teoremi.

Tecnologie utilizzate

JavaFX

JavaFX è un insieme di pacchetti graficia e multimediali che consente agli sviluppatori di progettare,
creare, testare, eseguire il debug e distribuire applicazioni rich client che operano in modo coerente nelle diverse piattaforme.
Dal momento che la libreria JavaFX è scritto come un API Java, il codice dell'applicazione JavaFX può fare riferimento ad API di qualsiasi libreria Java.
Ad esempio, le applicazioni JavaFX possono utilizzare librerie Java API per accedere a funzionalità native del sistema.
L'aspetto grafico delle applicazioni JavaFx èpersonalizzabile: si possono ad esempio usare fogli di stile (css) oppure utilizzare il linguaggio di scription fxml per separare la parte di layout grafico da quella di sviluppo delle funzionalita'.
Le Api JavaFx sono integrate nel Java SE Runtime Environment (JRE) e nel Java Development Kit (JDK).
Dato che JDK èdisponibile per la maggior parte delle piattaforme desktop (Windows, Mac OS e Linux), le applicazioni JavaFx compilate con JDK 8 possono essere eseguite sulla maggior parte delle piattaforme.

Architettura

Questo paragrafo fornisce una descrizione ad alto livello dell'architettura di JavaFx.

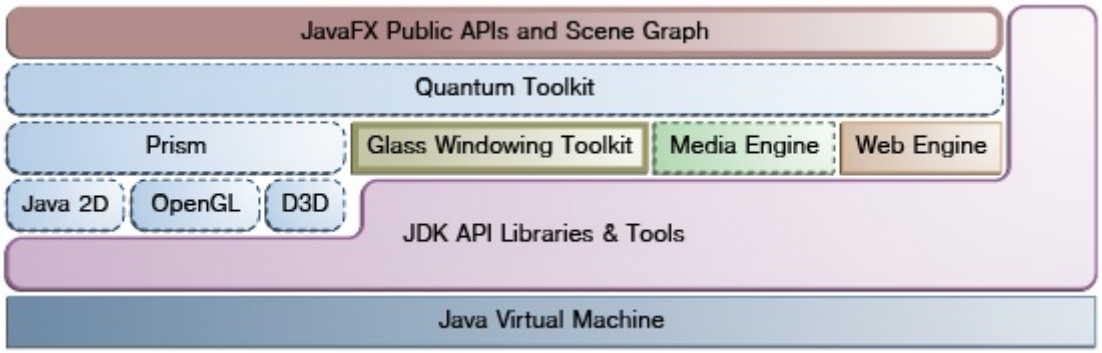


Illustrazione 20: Architettura JavaFX

Scene Graph

Scene graph, nella parte alta dell'architettura, è il punto dipartenza per costruire applicazioni JavaFx. È una gerarchia ad albero di nodi che rappresentano tutti gli elementi visivi dell'interfaccia utente.

Un singolo elemento nel scene graph è chiamato nodo.

Ogni nodo possiede un ID, una classe per lo stile e dimensioni limitate.

Fatta eccezione per il nodo principale dello scene graph (root node), ogni nodo ha un unico genitore e zero o piu' figli.

Ogni nodo possiede inoltre le seguenti caratteristiche:

- Effetti (blur, shadows)
- Opacita'
- Trasformazioni
- Gestori di eventi

Il JavaFx scene graph include inoltre primitive grafiche, come rettangoli e testo, in modo da avere container di layout, immagini e media

Le API `javafx.scene` permettono di creare e specificare diversi tipi di contenuti, come:

- Nodi : forme (2-D e 3-D), immagini, testi, browser web, grafici, gruppi, contenitori
- Stati: trasformazioni (posizionamento e orientamento dei nodi), effetti visivi ed altri stati visivi dei contenuti
- Effetti: semplici oggetti che cambiano l'aspetto del nodo, come l'effetto sfocato (blur), l'ombreggiatura (shadow) o le modifiche di colore

Java public APIs for JavaFX

Il livello piu' alto dell'architettura JavaFX contiene una set completo di api Java che supportano lo sviluppo di applicazioni rich client.

Queste APIs forniscono liberta' e flessibilita' nel costruire applicazioni rich client, dando a JavaFX le seguenti caratteristiche:

- permettono di poter utilizzare le piu' potenti caratteristiche Java, come i tipi generici, annotazioni, multithreading ed espressioni Lamda
- rendono semplice per uno sviluppatore web che utilizzano altri linguaggi basati su JVM come Groovy o Javascript usare JavaFX
- permettono ad uno sviluppatore Java di usare altri linguaggi di sistema, come Groovy, per scrivere ampie e complesse applicazioni JavaFX

- estende la libreria Java includendo le observable list e maps, che permettono alle applicazioni di collegare interfaccia utente a modelli dati, osservarne i cambiamenti e aggiornare l'interfaccia

Molte delle APIs JavaFX sono state incluse direttamente in Java. Alcumen APIs, come Layout e Media, sono state migliorate e semplificate basandosi sul feedback ricevuto dagli utenti della release di JavaFx 1.x.

Graphics System

La JavaFX Graphics System, si trova sotto il livello scene graph.

Supporta sia 2-D che 3-D ed èprovvisto di software per il rendering che iterviene quando l'hardware grafico del sistema èinsufficiente.

Ci sono due pipeline grafiche implementate nella piattaforma JavaFX:

- **Prism**, che puo' essere eseguito sia su software che hardware per il rendering, incluso il 3-D. Si occupa di effettuare il rendering delle JavaFx scenes.

  I render sono possibili in base al dispositivo in uso:

  - DirectX 9 su Windows XP e Windows Vista
  - DirextX 11 su Windows 7
  - OpenGL su Mac, Linux, Embedded
  - rendering software quando l'accelerazione hardware non èpossibile

l'accelerazione hardware è usata quando è possibile, ma quando non è disponibile,è usato il render software perché distribuito in tutte le Java Runtime Environments (JRE).

Glass Windowing Toolkit

Il glass windonwing toolkit, visualizzato in grigio al centro della figura 1.1, èil livello piu' basso nello stack grafico di JavaFx. La sua principale funzionalita' èquella di fornire un servizio operativo nativo, come gestire le finestre, i timer, etc.

Serve per connettere la piattaforma JavaFx al sistema operativo nativo.

Il glass windowing tollkit èanche responsabile della gestione della coda di eventi: utilizza la coda di eventi nativa del sistema operativo per definire l'uso dei thread.

Il glass toolkit èeseguito sullo stesso thread dell'applicazione JavaFx

Threads

Il sistema esegue uno o piu' dei seguenti thread in un dato momento

- **applicazione JavaFx** : èil thread primario utilizzato dagli sviluppatori di applicazioni JavaFx. Ogni scena “live”, che rappresenta una parte di una finestra, deve essere raggiungibile da questo thread. La scena grafica puo' essere creata e manipolata in un thread eseguito in background, ma quando il suo nodo root ècollegato ad un qualunque oggetto oggetto “attivo” nella scena, questa scena grafica deve essere raggiungibile dal thread dell'applicazione JavaFX. Questo permette agli sviluppatori di realizzare scene grafiche complesse in un thread in background, mantenendo le animazioni della scena “live” fluide.
- **Render Prism**: questo thread gestisce il rendering in modo indipendente dall gestore degli eventi. Permette di visualizzare ilframe N fino a che il frame N+1 viene processato. Questa abilita' di avere processi concorrenti èun grande vantaggio, specialmente sui moderni sistemi operativi che dispongono di multi-processore.
- **Media**: questo thread viene eseguito in background e sincronizza l'ultimo frame attraverso la scena grafica usanto il thread dell'applicazione JavaFX

Pulse

Un impulso (pulse) èun evento che indica alla scena grafica di JavaFX che èil momento di sincronizzare lo stato degli elementi sulla scena con Prism. L'impulso è limitato a 60 frames al secondo (fps) e viene emesso ogni volta che sulla scena sono in esecuzione animazioni. Anche se non ci sono animazioni, è schedulato un impulso quando qualcosa nella scena è cambiato, come ad esempio la posizione di un bottone.

Quando viene emesso l'impulso, lo stato degli elementi della scena grafica èsincronizzato fino al livello di render. Un impulso mette a disposizione degli sviluppatori un modo per gestire eventi in modo asincrono, permettendo al sistema di preparare ed eseguire eventi al momento dell'impulso.

Anche il layout e il CSS sono legati all'impulso. Numerosi cambiamenti nella scena grafica possono possono portare ad avere layout multipli o aggiornamenti CSS, con la possibilita' di degradare le performance.

Per mantenere le prestazioni, il sistema permette di lasciare passare la sincronizzazione di una sola modifica ad impulso.

Media and Images

Questa funzionalita' è disponibile attraverso l'API `javafx.scene.media`. Supporta sia la parte visuale che la parte multimediale, fornendo il supporto per MP3, AIFF, file audio WAV e file video FLV.

JavaFX Media è composta da tre componenti separate: Media gestisce i file multimediali, MediaPlayer che gestisce l'esecuzione del file e MediaView è il nodo si occupa di visualizzare il contenuto multimediale.

Il Media Engine, visualizzato nella figura 1.1, è stato progettato pensando a performance e stabilita' e fornisce un comportamento coerente fra le varie piattaforme.

Web Component

La componente Web è una UI JavaFX che fornisce un visualizzatore web e la piena funzionalità di navigazione attraverso la sua API.

Web Engine, visualizzato in arancio nella figura 1.1, è basato su Webkit, un motore browser web open source che supporta HTML5, CSS, JavaScript, DOM e SVG.

Questo permette agli sviluppatori di implementare le seguenti caratteristiche nelle loro applicazioni Java:

- Visualizzare HTML da contenuti locali o URL remoti
- Avere il supporto history che fornisce la navigazione Back / Forward
- Ricaricare i contenuti
- Applicare effetti alle componenti web
- Modificare il contenuto HTML
- Eseguire comandi JavaScript
- Gestire eventi

Questa componente browser integrata, è composta da due classi:

- `WebEngine` fornisce la capacità di navigazione web di base
- `WebView` incapsula un oggetto WebEngine, incorpora contenuti HTML in una scena dell' applicazione, e offre campi e metodi per applicare effetti e trasformazioni. Si tratta di un'estensione di una classe Node.

CSS

JavaFX Cascading Style Sheets (CSS) offre la possibilità di applicare uno stile personalizzato all'interfaccia utente di un'applicazione JavaFX senza modificare il codice sorgente.

Il CSS può essere applicato a qualsiasi nodo della scena grafica JavaFX e viene applicato ai nodi in modo asincrono. Stili JavaFX CSS possono anche essere facilmente assegnati alla scena in fase di runtime, consentendo di modificare dinamicamente l'aspetto di un'applicazione .

La figura 1.1 mostra la stessa applicazione con due stili CSS differenti sullo stesso set di controlli UI

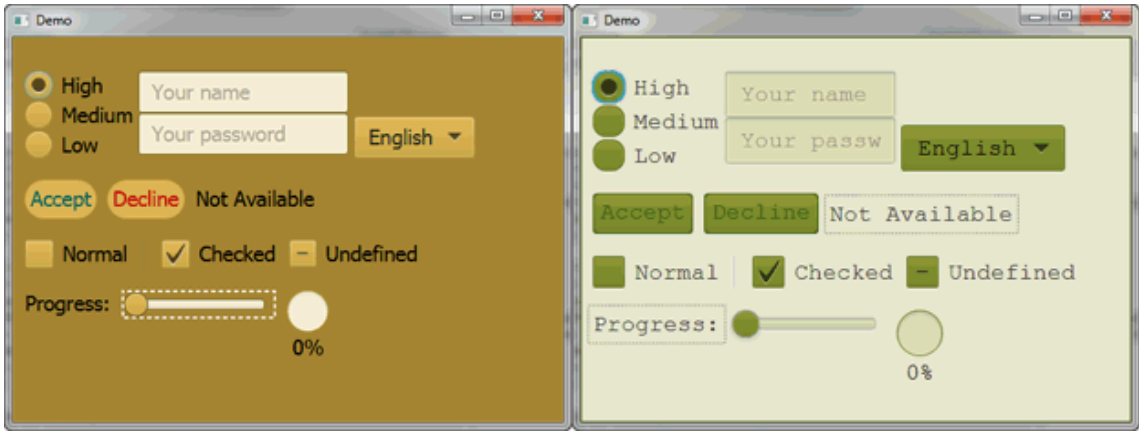


Figura 1.2 - esempio di CSS

JavaFX CSS è basato sulle specifiche del CSS W3C versione 2.1, con alcune aggiunte dal lavoro in corso sulla versione 3. Il supporto JavaFX CSS e le estensioni sono state progettate per consentire ai fogli di stile CSS per JavaFX di essere analizzati in modo pulito da qualsiasi parser CSS compliant, anche uno che non supporta le estensioni JavaFX. Ciò consente la miscelazione di stili CSS per JavaFX e per altri scopi (ad esempio per le pagine HTML) in un unico foglio di stile. Tutti i nomi di proprietà JavaFX sono precedute da una estensione "-fx-", comprese quelle che potrebbero sembrare compatibile con CSS HTML standard, perché alcuni valori JavaFX hanno leggermente diversi semantiche.

UI Controls

I controlli JavaFX UI disponibili attraverso l'API JavaFX sono costruiti utilizzando nodi

nella scena grafica. Possono sfruttare appieno le caratteristiche visivamente ricche di JavaFX e sono portabili su diverse piattaforme.

Questi controlli risiedono nel package `javafx.scene.control`.

Layout

I contenitori layout o riquadri possono essere utilizzati per visuallizzare i controlli UI entro una scena grafica di un'applicazione JavaFX. L'API layout JavaFX include le seguenti classi contenitore che automatizzano i modelli di layout comuni:

- `BorderPane` espone i suoi nodi contenuti in alto, in basso, a destra, a sinistra o al centro
- `HBox` organizza suoi nodi contenuti orizzontalmente in una singola fila
- `VBox` organizza suoi nodi contenuti verticalmente in una singola colonna
- `StackPane` pone suoi nodi di contenuti in un unico stack back-to-front
- `GridPane` permette allo sviluppatore di creare una griglia flessibile di righe e colonne in cui disporre i nodi
- `FlowPane` organizza i suoi nodi in un "flusso" orizzontale o verticale , aumentando le dimensioni fino alla larghezza specificata (orizzontale) o altezza (verticale) entro i confini indicati
- `TilePane` colloca i suoi nodi in layout uniforme
- `AnchorPane` consente agli sviluppatori di creare nodi di ancoraggio verso l'alto, in basso, lato sinistro o al centro del layout

Per realizzare una struttura di layout desiderata, diversi contenitori possono essere nidificati all'interno di un'applicazione JavaFX.

Trasformazioni 2-D e 3-D

Ogni nodo del grafo della scena JavaFX può essere trasformato in coordinate xy utilizzando le seguenti classi `javafx.scene.tranform`:

- `translate` – per spostare un nodo da un luogo all'altro lungo x, y, z piani relativi alla sua posizione iniziale.
- `scale` – per ridimensionare un nodo per apparire più grande o più piccolo nelle

direzioni x, y, z piani, a seconda del fattore di scala.

- `shear` – per ruotare un asse in modo che l'asse X e l'asse Y sono più perpendicolari. Le coordinate del nodo vengono spostati dai moltiplicatori specificati
- `rotate` – per ruotare un nodo di un punto di articolazione specificato della scena.
- `affine` – per eseguire una mappatura lineare da coordinate 2-D / 3-D coordinate di altri 2-D / 3-D mantenendo le proprietà di 'drittè e «parallelo» delle linee. Questa classe deve essere usato con `translate`, `scale`, `rotate` o `shear` invece di essere utilizzato direttamente

Effetti Visivi

Lo sviluppo di interfacce rich client JavaFX implica l'uso di effetti visivi o effetti per migliorare l'aspetto delle applicazioni JavaFX in tempo reale. Il JavaFX Effects sono principalmente pixel dell'immagine-based e, di conseguenza, prendono l'insieme di nodi che sono nella scenagrafica, la renderizzano come immagine, ed applicano gli effetti su di essa.

Alcuni degli effetti visivi disponibili in JavaFX includono l'impiego delle seguenti classi:

- `Drop Shadow` – renderizza l'ombra di un dato contenuto
- `Reflection` – renderizza un riflesso del contenuto
- `Lighing` - simula una fonte di luce che brilla su un dato contenuto e può dare ad un oggetto piatto un aspetto tridimensionale più realistico